Framework for Implementation of Iterative Learning Control on Programmable Logic Controllers

Matthias Bibl, Michael Robin, Michael Steinegger, and Georg Schitter

Automation and Control Institute, Vienna University of Technology
Gußhausstraße 27-29, A-1040 Vienna, Austria
Email: {bibl,robin,steinegger,schitter}@acin.tuwien.ac.at

***Abstract*—In this paper, an implementation approach for norm-optimal iterative learning control (ILC) on programmable logic controllers (PLCs) is presented. After a detailed conceptual overview and discussion of the norm-optimal ILC algorithm, the challenges for implementing ILC algorithms on PLCs are discussed and an efficient three-phase implementation approach is proposed. Here, the three phases consist of an offline calculation, the calculation of the feedforward part between consecutive iterations, and the online calculation of the current control input. It is also shown that this separation enables the efficient implementation of the norm-optimal ILC algorithm on standard industrial controllers like PLCs. The proposed norm-optimal ILC implementation approach is verified by a simulation of a gantry robot with three degrees of freedom, where the norm-optimal ILC algorithm is executed within a Soft-PLC.**

I. Introduction

Handling systems and industrial robots are often applied in the manufacturing domain for repetitive tasks to significantly increase throughput and accuracy. However, repeating disturbances or gradually changing parameters like friction, gear backlash, or temperature drift makes precise trajectory tracking a challenging task and often learning control methods are applied [1]. Due to the repetitive nature of typical manipulator tasks it is possible to learn from previous motion cycles, and thus compensate for repetitive disturbances. *Iterative learning control* (ILC) is a well-known method to compensate for such disturbances and is summarized, e.g., by Bristow et al. [1]. Owens [2] also discussed two common ILC approaches: the norm-optimal ILC and the inversion-based ILC approach.

Considering prevalent handling systems in industry from a flexible production point of view, two main challenges have been identified. On the one hand, the control hardware of almost all manufacturers of industrial robotic systems is vendor-specific and highly coupled with the delivered robot. This makes the application of alternative control code and hardware almost impossible. Furthermore, the programming of specific tasks for the robotic system is often only possible by vendor-specific software or specific programming languages [3].

To enable a high flexibility of handling systems in modern production lines, new approaches that enable the possibility for flexible combination of plant components, also including the control system itself, are required [4]. According to Wicks [3], a shared control architecture between the robotic system and other hardware components applied for the manufacturing process may significantly reduce the system complexity. By robot control based on *programmable logic controllers* (PLCs), common programming languages and software interfaces can be used, which reduce engineering time and maintenance costs. In this case, however, new concepts for running more complex control algorithms efficiently on PLCs are required.

978-1-5090-1314-2/16/$31.00 © 2016 IEEE

In this paper, an approach for implementing a norm-optimal ILC method on a standard PLC is proposed. The algorithm described by Schindele and Aschmann [5] is partitioned into three parts: (i) offline calculations, (ii) calculation of the feedforward term for the next iteration between iterations, and (iii) online calculation of the current control input. This partition enables the efficient implementation on a standard industrial controller and reduces the implementation complexity of the algorithm since the individual parts can be calculated separately.

The remainder of the paper is organized as follows. In Sec. II the norm-optimal ILC algorithm is derived, whose implementation method is then described in Sec. III. Simulation results of the obtained norm-optimal ILC algorithm are presented in Sec. IV and Sec. V concludes the paper.

II. Norm-Optimal Iterative Learning Control

Fast and precise tracking of predefined trajectories is one of the fundamental tasks that have to be solved in robotics. Considering the dynamics of a rigid-body-system given as

$$\boldsymbol{\tau} = \boldsymbol{\mathcal{M}}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{\mathcal{C}}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) \ , \tag{1}$$

where n is the number of degrees of freedom, $\boldsymbol{\tau} \in \mathbb{R}^{n\times 1}$ are the generalized forces for the fully actuated system, $\boldsymbol{q} \in \mathbb{R}^{n\times 1}$ the generalized joint variable vector, $\boldsymbol{\mathcal{M}}(\cdot) \in \mathbb{R}^{n\times n}$ the inertia matrix, $\boldsymbol{\mathcal{C}}(\cdot) \in \mathbb{R}^{n\times n}$ represents the matrix of coriolis forces, and $\boldsymbol{g}(\cdot) \in \mathbb{R}^{n\times 1}$ is the gravity vector. It can be seen that a sequence of desired torques $\boldsymbol{\tau}^{\text{des}}$ has to be applied to the system of the form (1) in order to follow a desired trajectory.

The non-linear system dynamics (1) can be transformed to a *linear time-invariant* (LTI) state space model of the form

$$\dot{\boldsymbol{x}}_j = \boldsymbol{A}\boldsymbol{x}_j + \boldsymbol{B}\boldsymbol{u}_j \tag{2a}$$
$$\boldsymbol{y}_j = \boldsymbol{C}\boldsymbol{x}_j \tag{2b}$$

by feedback linearization techniques (cf. Slotine and Li [6]). Here, $\boldsymbol{x} \in \mathbb{R}^{n\times 1}$ represents the system states, $\boldsymbol{u} \in \mathbb{R}^{m\times 1}$ the system input with m being the number of system inputs, $\boldsymbol{y} \in \mathbb{R}^{p\times 1}$ the system output with p being the number of system outputs, $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is the system matrix, $\boldsymbol{B} \in \mathbb{R}^{n\times m}$ is the input matrix, and $\boldsymbol{C} \in \mathbb{R}^{p\times n}$ is the output matrix. The index j represents herein the iteration index of a repetitive trajectory. However, the control implementation for the LTI system (2) on a standard industrial controller like a PLC requires a discrete-time system representation given as

$$\boldsymbol{x}_j[k+1] = \boldsymbol{\Phi}\boldsymbol{x}_j[k] + \boldsymbol{\Gamma}\boldsymbol{u}_j[k] \tag{3a}$$
$$\boldsymbol{y}_j[k] = \boldsymbol{\Xi}\boldsymbol{x}_j[k] \tag{3b}$$

where $k \in \mathbb{Z}, (0 \leq k \leq N)$ is the index of the sampling point, N is the sample length, $\boldsymbol{\Phi} \in \mathbb{R}^{n \times n}$ the discrete system matrix, $\boldsymbol{\Gamma} \in \mathbb{R}^{n \times m}$ the input matrix, and $\boldsymbol{\Xi} \in \mathbb{R}^{p \times n}$ is the output matrix of the introduced discrete-time system representation (3).

A. Control Concept

For readability reasons and without loss of generality, the norm-optimal ILC control concept is described for systems with one degree of freedom ($m = 1$). According to Schindele and Aschemann [5], the goal of the norm-optimal ILC approach is to minimize the quadratic cost function J defined as

$$J = \frac{1}{2}\left(\boldsymbol{e}_{j+1}^{\mathrm{T}}\boldsymbol{S}\boldsymbol{e}_{j+1} + (\boldsymbol{v}_{j+1} - \boldsymbol{v}_j)^{\mathrm{T}}\boldsymbol{R}(\boldsymbol{v}_{j+1} - \boldsymbol{v}_j)\right) , \quad (4)$$

which has to be minimized for the next control input $\boldsymbol{v}_{j+1}$. Here, $\boldsymbol{e} \in \mathbb{R}^{N \times 1}$ is the output error vector, which contains all individual error samples between the desired system output $\boldsymbol{y}_{j+1}^{\mathrm{des}}[k]$ and the previous output $\boldsymbol{y}_j[k]$ for the subsequent iteration $j+1$. The matrices $\boldsymbol{S} \in \mathbb{R}^{N \times N}$ and $\boldsymbol{R} \in \mathbb{R}^{N \times N}$ are symmetric and positive definite weighting matrices and $\boldsymbol{v} \in \mathbb{R}^{N \times 1}$ is the sampled control input vector. In order to solve this optimization problem, the hamiltonian function H is applied as

$$\begin{aligned} H = &-\frac{1}{2}\boldsymbol{e}_{j+1}^{\mathrm{T}}[k]\boldsymbol{S}[k,k]\boldsymbol{e}_{j+1}[k] \\ &-\frac{1}{2}\left(\boldsymbol{v}_{j+1}[k] - \boldsymbol{v}_j[k]\right)^{\mathrm{T}}\boldsymbol{R}[k,k]\left(\boldsymbol{v}_{j+1}[k] - \boldsymbol{v}_j[k]\right) \\ &+\boldsymbol{\lambda}_{j+1}^{\mathrm{T}}[k+1]\left(\boldsymbol{\Phi}\boldsymbol{x}_{j+1}[k] + \boldsymbol{\Gamma}\boldsymbol{v}_{j+1}[k]\right) \end{aligned} \quad (5)$$

where $\boldsymbol{\lambda}_{j+1} \in \mathbb{R}^{n \times 1}$ denotes the so-called co-state. According to Athans and Falb [7], the optimal solution can then be obtained by solving the following three equations given as

$$\frac{\partial H}{\partial \boldsymbol{\lambda}_{j+1}[k+1]} = \boldsymbol{x}_{j+1}[k+1] , \quad (6a)$$

$$\frac{\partial H}{\partial \boldsymbol{x}_{j+1}[k]} = \boldsymbol{\lambda}_{j+1}[k] , \quad (6b)$$

$$\frac{\partial H}{\partial v_{j+1}[k]} = 0 . \quad (6c)$$

It has been shown in [5], that by defining the co-state as

$$\boldsymbol{\lambda}_{j+1}[k] = -\boldsymbol{P}[k]\Delta\boldsymbol{x}[k] + \boldsymbol{\xi}_{j+1}[k] , \quad (7)$$

with $\Delta\boldsymbol{x}[k] = \boldsymbol{x}_{j+1}[k] - \boldsymbol{x}_j[k]$, the matrix $\boldsymbol{P} \in \mathbb{R}^{n \times n}$, and the vector $\boldsymbol{\xi}_{j+1} \in \mathbb{R}^{n \times 1}$, the solution for the input of the next iteration $\boldsymbol{v}_{j+1}$ results in the equation

$$\boldsymbol{v}_{j+1}[k] = \boldsymbol{v}_j[k] - \left(\boldsymbol{\Psi}\left(\boldsymbol{P}[k+1]\boldsymbol{\Phi}\Delta\boldsymbol{x}[k] - \boldsymbol{\xi}_{j+1}[k+1]\right)\right) \quad (8)$$

with $\boldsymbol{\Psi} = \left(\boldsymbol{\Gamma}^{\mathrm{T}}\boldsymbol{P}[k+1]\boldsymbol{\Gamma} + \boldsymbol{R}[k,k]\right)^{-1}\boldsymbol{\Gamma}^{\mathrm{T}}$. Therein, the optimal feedforward term $\boldsymbol{\xi}_{j+1}[k]$ defined as

$$\begin{aligned} \boldsymbol{\xi}_{j+1}[k] = &\boldsymbol{\Phi}^{\mathrm{T}}\left(\boldsymbol{I} - \boldsymbol{P}[k+1]\boldsymbol{\Gamma}\boldsymbol{\Psi}\right) \\ &\boldsymbol{\xi}_{j+1}[k+1] + \boldsymbol{\Xi}^{\mathrm{T}}\boldsymbol{S}[k,k]\boldsymbol{e}_j[k] , \end{aligned} \quad (9)$$

where $\boldsymbol{I} \in \mathbb{R}^{n \times n}$ is the identity matrix and $\boldsymbol{P}$ is a solution of the Riccati equation. The terminal conditions are given as

$$\boldsymbol{\xi}_{j+1}[N] = \boldsymbol{\Xi}^{\mathrm{T}}\boldsymbol{V}[N,N]\boldsymbol{e}_j[N] \quad (10a)$$

$$\boldsymbol{P}[N] = \boldsymbol{\Xi}^{\mathrm{T}}\boldsymbol{V}[N,N]\boldsymbol{\Xi} . \quad (10b)$$

In the following section, several guidelines for reducing the computational complexity and implementing the described norm-optimal ILC algorithm on a standard PLC are derived.

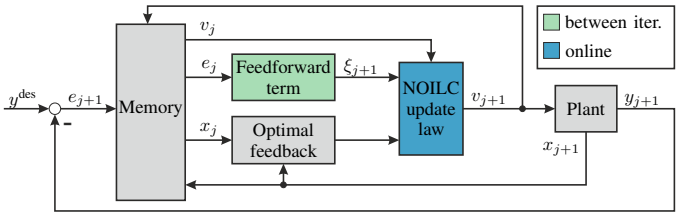

Fig. 1. Block diagram of the norm-optimal iterative learning control algorithm based on Schindele and Aschemann [5]. The online phase and the calculation between iterations for the implementation on a PLC are marked with color.

III. Guidelines for Implementation

The norm-optimal ILC algorithm described in Sec. II can also be written in the standard ILC representation given as

$$\boldsymbol{v}_{j+1} = \boldsymbol{Q}\left(\boldsymbol{v}_j + \boldsymbol{L}\boldsymbol{e}_j\right) , \quad (11)$$

with an additional state term. Here, $\boldsymbol{v}$ and $\boldsymbol{e}$ are represented in the so-called super-vector description (cf. Bristow et al. [1]), $\boldsymbol{Q} \in \mathbb{R}^{N \times N}$ is a filter matrix to suppress higher frequent errors due to model uncertainties and $\boldsymbol{L} \in \mathbb{R}^{N \times N}$ is the learning gain. Applying a shared PLC-based control architecture between a robotic system and additional components of a production line requires the integration of eventually required computationally complex control algorithm like the ILC algorithm on a PLC.

A. Efficiency Analysis

The implementation of the ILC law (11) with standard linear algebra techniques requires $N \times (N-1) \times N + N$ operations, which results in a computational complexity of $\mathcal{O}(n^3)$. Assuming an exemplary trajectory with a motion cycle time of 1 s and a sampling time of 1 ms leads to a number of $N = 1000$ samples. Executing the update law (11) for this trajectory on a hardware PLC, like a Siemens SIMATIC S7-300 whose execution time for a single floating-point operation is 590 ns, takes $\approx$ 590 s or around 10 min to compute the new control input $\boldsymbol{v}_{j+1}$. This execution time is not feasible for industrial processes like pick and place operations with a typical cycle time in the range of a few seconds. Thus, a more efficient implementation approach is required to enable the execution of the ILC algorithm on standard industrial controllers.

B. Partitioning of the Algorithm

According to Ratcliffe et al. [8], the norm-optimal ILC update law (8) and the feedforward term (9) can be rewritten as

$$\boldsymbol{v}_{j+1}[k] = \boldsymbol{v}_j[k] - \boldsymbol{\alpha}[k]\Delta\boldsymbol{x}[k] + \boldsymbol{\beta}[k]\boldsymbol{\xi}_{j+1}[k] \quad (12)$$

with $\boldsymbol{\xi}_{j+1}[k]$ being defined as

$$\boldsymbol{\xi}_{j+1}[k] = \boldsymbol{\gamma}[k]\boldsymbol{\xi}_{j+1}[k+1] + \boldsymbol{\omega}\boldsymbol{e}_j[k] , \quad (13)$$

where the individual parts are given as

$$\boldsymbol{\alpha}[k] = \boldsymbol{\Psi}\boldsymbol{P}[k+1]\boldsymbol{\Phi} , \quad (14a)$$

$$\boldsymbol{\beta}[k] = \boldsymbol{\Psi} , \quad (14b)$$

$$\boldsymbol{\gamma}[k] = \boldsymbol{\Phi}^{\mathrm{T}}\left(\boldsymbol{I} - \boldsymbol{P}[k+1]\boldsymbol{\Gamma}\boldsymbol{\Psi}\right) , \quad (14c)$$

$$\boldsymbol{\omega}[k] = \boldsymbol{\Xi}^{\mathrm{T}}\boldsymbol{S}[k,k] . \quad (14d)$$

By applying the partition of the update law according to (12) togehter with (13) and (14) enables the division of the entire ILC algorithm into three phases: (i) offline phase, (ii) calculations between iterations, (iii) and online phase.

1) Offline phase: The offline phase consists of computing $\boldsymbol{\alpha}[k], \boldsymbol{\beta}[k], \boldsymbol{\gamma}[k]$, and $\boldsymbol{\omega}[k]$ in (14) after the model of the desired plant to be controlled was transformed to the system representation (3). In this phase, the main computational effort is caused by solving the Ricatti equation in order to obtain the solution for $\boldsymbol{P}$. Since the offline calculations are not time critical, most parts of the calculations could also be carried out on a PLC with proper implementation of linear algebra methods. However, since the computational power of the PLC could be required for other control tasks in a shared control architecture, the calculations of (14) should be carried out in a well-suited environment like, for example, MATLAB. The results can then be transferred to the PLC and used to compute the feedforward term (13) in the subsequent phase.

2) Calculations between iterations: Considering, for instance, a pick and place task of a handling system, the entire end-effector trajectory can be split into two parts. The first part is the movement from the initial end-effector position (pick position) to the desired pose where the handled component should be placed. After reaching the desired place position, the end-effector has to be moved back to the initial or next pick position. Thus, the goal of the ILC is to learn the trajectory of the first movement part while the second part can be achieved by using standard feedback controllers. The time needed for moving the end-effector back to the starting position can then be used to compute the results of (13). Note that (13) is calculated in a backward manner, which requires the previous ILC cycle to be finished and, therefore, the calculation cannot be executed earlier. Here, the time t_ξ required for calculating (13) depends on the number of samples N and the number of system outputs. In order to evaluate the possible computation of (13) within one backward movement cycle, the computation time t_ξ has to be compared with the backward movement cycle time t_b. If $t_\xi > t_b$, then two possible solutions can be considered in order to still execute the ILC algorithm on the PLC. The obvious solution is to wait for the computation of (13) to be finished, which is often not a feasible approach since this would increase the production time. The second solution is to apply the ILC control input from the previous movement and then apply the new calculated input for the subsequent iteration when the calculation is done.

3) Online phase: The remaining calculations of the ILC algorithm for the next iteration is to compute the update vector according to (12) for each iteration sampling point k. Since the results of (13) and (14) have already been calculated either offline or during the backward movement phase, the calculation of the update law is not computationally complex since it requires only simple vector-matrix computations. Here, the number of operations for each sampling point only depends on the number of manipulator states and not on the iteration length as the calculations between iterations.

C. Additional Considerations

The proposed algorithm already includes a norm-optimal feedback controller (see Fig. 1, optimal feedback block). Therefore the design of a separate feedback controller and including it into the ILC algorithm can be omitted [5].

The feedback term in the update law in (12) depends on the full system state $\boldsymbol{x}$. Thus, the underlying system is required to be fully observable. For handling or robotic systems this mostly encompasses the position and velocity, which are usually measured by encoders included in the driving motors.

In (11), $\boldsymbol{Q}$ is typically designed as a low-pass filter to suppress higher frequent errors due to model uncertainties and measurement noise. In (12), it can be seen that $\boldsymbol{Q}$ equals the identity matrix $\boldsymbol{I}$ in this case. Thus, an additional Savitzky-Golay-Filter [9] has been added and implemented. The filter calculates the best approximation of a certain order polynomial for each point regarding a certain window size. This introduces a smooth transition for the next input.

IV. Evaluation

In order to evaluate the ILC implementation approach proposed in Sec. III, a simulation on a PLC is carried out.

A. System Description

The evaluation system consists of a Soft-PLC from Beckhoff Automation (TwinCAT, Version 3.1.4018.4, 1 ms cycle time). The targeted plant is an H-bridge three axis (X, Y and Z) portal system as shown in Fig. 2, where currently only the X-axis is simulated for evaluation of the ILC implementation on the Soft-PLC. As model of the plant a Simulink-model of a mass-spring-damper system, as described in [10, p. 198], is used and integrated in TwinCAT using the specific compiler Add-on developed by Beckhoff. The estimated and used parameters for the hardware model of the X-axis are $m = 5\,\mathrm{kg}, k = 0\,\mathrm{N/m}$ and $d = 0.01\,\mathrm{Ns/m}$. For an easier usability, an interface in Microsoft Visual Studio using C# to enter parameters has been programmed. The offline-calculations of the parameters in (8) and (9) are carried out in MATLAB and the results are transferred to the Soft-PLC via the ADS-protocol of TwinCAT.

The ILC algorithm starts with the parameters $m = 2\,\mathrm{kg}$, $k = 0\,\mathrm{N/m}$ and $d = 0\,\mathrm{Ns/m}$ to simulate a mismatch between the model and the simulated system. To simulate sensor noise a pseudo-random number is calculated and scaled such that the RMS-value of the position noise is $29\,\mu\mathrm{m}$ and the RMS-value of the velocity noise is $29\,\mu\mathrm{m/s}$ which is in the order of the encoder resolution of the system in Fig. 2.

B. Simulation Results

The planned trajectory for the X-axis (double axis in Fig. 2) is from 0 mm to 500 mm with a duration of 2 s. The trajectory

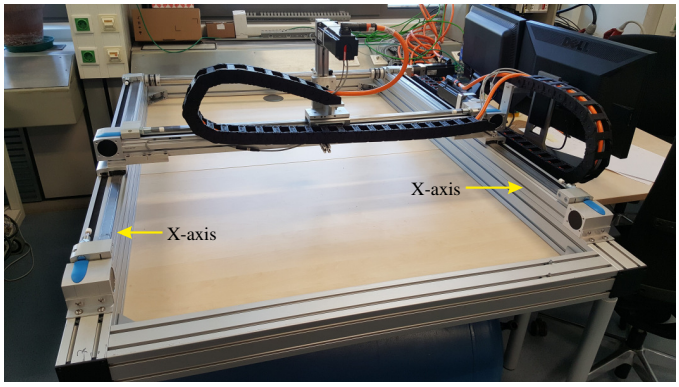

Fig. 2. For the simulation model a sub-part of a three axis H-bridge system is applied. The system consists of three translational axis (X, Y and Z) which are stacked in a serial kinematic manner. For evaluation purposes, only the X-axis was simulated on the Soft-PLC and cross-coupling effects are neglected due to presentation reasons in this publication.

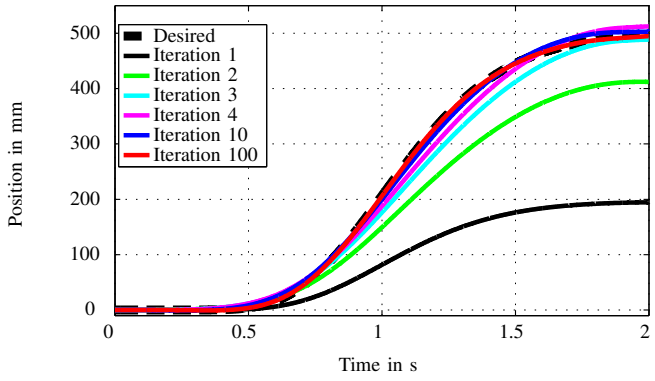

Fig. 3. The control task consists of moving the X-axis from 0 mm to 500 mm within 2 s. The resulting trajectory for the first iteration (solid black line) shows a high deviation from the desired trajectory since the first iteration is pure feed-forward with the chosen wrong plant parameters. In total, 100 iterations are carried out which are not all shown in the figure. It can be seen, that the learning rate decreases from iteration to iteration.

is calculated such that a certain velocity, acceleration and jerk is not exceeded. In total 100 iterations with active ILC are carried out before the simulation is stopped.

In Fig. 3 a few selected iterations of the ILC algorithm are shown. The trajectory for the first iteration (solid black line) shows very poor results. The reasons for this is that there is no current error value available for the first iteration and the plant parameters have been chosen different from the correct values as described in the previous section. After the first iteration, the error between the trajectory of the previous iteration and the desired trajectory can be calculated and the ILC starts the learning process. It can also be seen in Fig. 3 that the learning rate decreases continuously from iteration to iteration.

Fig. 4 shows the change of the system input (solid lines) over the iterations. The corresponding feedback part of the system input is drawn in the same color but as dashed line. It can clearly be seen, that the feedback part is decreasing over iterations since the ILC learns the correct model parameters.

In order to evaluate the accuracy of the learned trajectories, the RMS-value of the error between the desired and the simulated trajectories in Fig. 3 is calculated over the full iteration length. Fig. 5 shows the RMS-error of all 100 iterations. At the 11^{th} iteration the RMS-error is 6.24 mm, after 50 iterations the

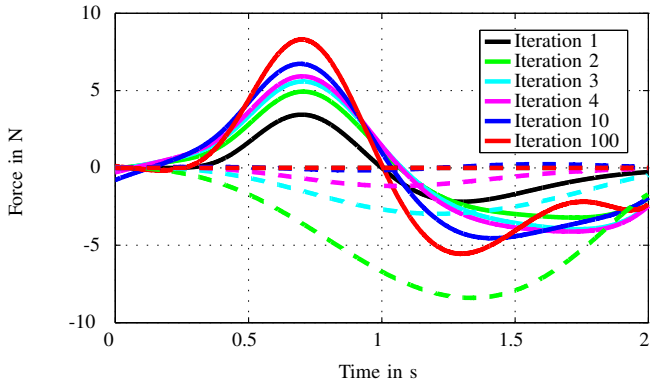

Fig. 4. Change of the system input v_j (solid lines) over the iterations and its feedback controller part $-\boldsymbol{\alpha}[k]\Delta\boldsymbol{x}[k]$ (dashed in the same color) from (12).

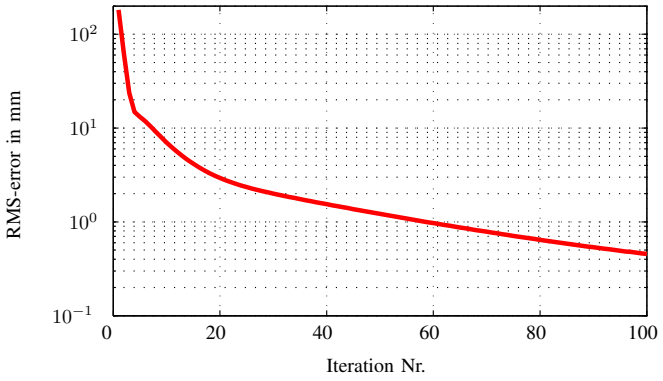

Fig. 5. The calculated RMS-error of all 100 iterations (logarithmic scale).

RMS-error decreases to 1.19 mm, and after 100 iterations the RMS-error over the entire trajectory is reduced to 0.45 mm.

V. Conclusion

This work shows how a norm-optimal ILC algorithm can be implemented on a PLC for a rigid-body-system. The algorithm is divided in (i) offline phase, (ii) calculations between iterations, and (iii) online phase. To show the convergence of the algorithm a three axis H-bridge model is chosen and a norm-optimal ILC for this system is implemented using a Soft-PLC. It is shown by simulation that even with a very error-prone model the algorithm is able to follow the desired trajectory with a small error of <1 mm over the entire trajectory after 60 iterations. Here, the calculations between iterations are finished within one backward movement cycle.

Based on this simulation results, current work is focused on the implementation on the real plant as shown in Fig. 2.

Acknowledgment

Financial support of the Austrian Research Promotion Agency (FFG) under grant no. 848623 is gratefully acknowledged.

References

[1] A. D. Bristow, M. Tharayil, and A. G. Alleyne, "A survey of iterative learning control – A learning-based method for high-performance tracking control," *IEEE Control Systems Magazine*, pp. 96–114, 2006.

[2] D. H. Owens, *Iterative Learning Control: An Optimization Paradigm*. Springer-Verlag, 2015.

[3] M. Wicks, "PLC-based robotic controls versus OEM robotic controls," *Intelligrated – White paper*, pp. 1–4, 2012.

[4] M. G. Mehrabi, A. G. Ulsoy, Y. Koren, and P. Heytler, "Trends and perspectives in flexible and reconfigurable manufacturing systems," *Journal of Intelligent Manufacturing*, vol. 13, pp. 135–146, 2002.

[5] D. Schindele and H. Aschemann, "Norm-optimal iterative learning control for a pneumatic parallel robot," in *Multibody System Dynamics, Robotics and Control*, H. Gattringer and J. Gerstmayr, Eds. Springer-Verlag Wien, 2013, ch. 7, pp. 113–128.

[6] J.-J. E. Slotine and W. Li, *Applied Nonlinear Control*. Prentice-Hall, 1991.

[7] M. Athans and P. Falb, *Optimal Control: An Introduction to the Theory and Its Applications*. Dover Publications, Inc., 1966.

[8] J. Ratcliffe, L. van Duinkerken, P. Lewin, E. Rogers, J. Hatonen, T. Harte, and D. Owens, "Fast norm-optimal iterative learning control for industrial applications," in *Proceedings of the 2005, American Control Conference, 2005.*, June 2005, pp. 1951–1956 vol. 3.

[9] A. Savitzky and M. J. E. Golay, "Smoothing and differentiation of data by simplified least squares procedures," *Analytical Chemistry*, vol. 36, no. 8, pp. 1627–1639, 1964.

[10] R. M. Schmidt, G. Schitter, A. Rankers, and J. Van Eijk, *The Design of High Performance Mechatronics: High-Tech Functionality by Multidisciplinary System Integration*. IOS Press, 2011.